View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Elsevier - Publisher Connector

Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 162 (2006) 79–85

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

On the Design Rationale of ACP Style Process Algebras

Jan A. Bergstra[a,b,1]

[a] *University of Amsterdam, Faculty of Science, Programming Research Group, The Netherlands*

[b] *Utrecht University, Department of Philosophy, Applied Logic Group, The Netherlands*

Abstract

A review is given of the design rationale for ACP style process algebras. An outline of future directions of development within the given paradigm is given.

Keywords: Process algebra, ACP, equational presentation

1 Equational specifications for process algebras

This is not a historic paper in any sense and on purpose no attempt is made to trace facts, developments or remarks back to the literature. These connections and references are in fact easy to find for anyone who makes an attempt to do so.

The design of ACP (1984) as an equational presentation of process algebras emerged from several sources and technically stands in the tradition of the algebraic specification of abstract data types as it has been studied from the late sixties onwards by various research groups.

Right from the start of the development of ACP the following options and objectives for theoretical development have played a central role:

- The theory would be like formal language theory (Kleene algebra) in nature, but more general by removing one of its laws: the right distributivity of alternative composition over sequential composition. Sequential composition and alternative composition have been taken from formal language theory as basic combinators for behavior. This decision has a defining influence on the outcome of all further design stages. At the same time it is a parameter for process algebra design

[1] Email: janb@science.uva.nl

1571-0661 © 2006 Elsevier B.V. Open access under CC BY-NC-ND license.

doi:10.1016/j.entcs.2005.12.105

and different decisions made at this stage lead to other concurrency theories, comparable to but technically outside the ACP line.

- Axiom systems are designed in such a way that the concept of a model always can be found in preexisting logical theory. In fact it always suffices to consider axiom systems as theories in some classically known logic, and in most cases that is some fragment of first order logic with Tarski semantics as its concept of model.
- The design of chains (or rather trees) of enrichments of equational theories that provide an incremental number of features such as BPA, PA, ACP, ACP with state operator, or the same specifications each extended with a deadlocked step (delta), a silent step (tau) or an empty step (epsilon) or both of them. Feature interactions would be avoided by considering collections of features that mathematically coexist without difficulties. For instance combining the priority operator with weak bisimulation turned out to be difficult which gave rise to orthogonal bisimulation around 1998. These chains of equational theories have been inspired by the mathematical theories of quantities and numbers: groups, rings, fields, skew fields and so on. For each family of features (always including the BPA features of sequential and alternative composition) three design objectives would stand out first:
  · At first a process algebra that captures (strong) bisimulation semantics would be designed.
  · Equations must be such that all but sequential composition and alternative composition can be eliminated on finite terms. (The whole subject finds its roots in the objective to find elimination results for parallel composition using weakest axioms for that objective.)
  · Variable binding mechanisms are avoided if possible and sometimes at significant costs because these depart from the universal algebra background which constitutes the cornerstone of equational specifications of abstract data types.
- Each particular set of features (operators) is given an initial algebra specification by means of equations or when needed conditional equations. Preferably these equations are organized in such a way as to be useful as a term rewriting system as well. Whenever possible a deliberate effort is made to design and use finite sets of equations, and if needed auxiliary functions (as they are known in the theory of abstract data type specification) are designed. Demonstrating that such auxiliary operators are indeed a necessity has proven possible in a number of cases. It clearly is far harder to derive such negative information than to design the auxiliary operator for its specific purpose. The search for finitary specifications has led to operators such as left merge (and sometimes right merge) and communication merge, the unless operators used to specify the priority operators and a rich collection of auxiliary operators for timed versions of ACP.
- Making explicit use of different homomorphic images of initial algebras one obtains a range of semantic models for the features that are specified by a particular specification. Additional equations may be introduced to characterize such homomorphic images.

- Concrete and abstract features are distinguished, where abstract features in some form abstract from activity while concrete process algebras permit the counting of each step. For concrete process algebras models can be found using projective limit constructions applied to approximation algebras that are found as homomorphic images of the initial algebra for finite process by cutting off each process after a fixed number of steps. This projective limit construction provides the simplest semantic model for most of the ACP style process algebra specifications. The projective limits can be understood using initial algebra semantics and their approximation algebras and requires no excursion to either (structured) operational semantics or any bisimulation definitions. It can then be proved that bisimulation models based on SOS are in fact equivalent. I consider the projective limit model construction as the primary source of intuition on process equivalence in the case of concrete process algebras because of its extremely robust nature. It may work in cases where appropriate bisimulation definitions may be hard to develop due to a complex combination of features.

It has always been difficult to characterize exactly what it is that ACP style process algebra captures. This turns out to be a subjective matter to some extent. My own view is that ACP style process algebras intend to tell the (or rather a) story of processes in the classical format of universal algebra and equational logic. In addition the interpretation of alternative composition is of substantial importance because the very meaning of an alternative may vary from context to context. Ranging from the purely internal choice of an autonomous agent to the purely external choice felt by a key board sensing its user, many forms of more or less constrained choice are in between and ACP style process algebras should constitute a medium where this range of mechanisms of alternative composition can to some degree coexist within well-chosen models of axiom systems. As an example the treatment of fair abstraction can be mentioned where the fact that a choice may be infinitely often repeated to contributes to its meaning.

2 Further developments

Since its introduction ACP style process algebra has been developed in a number of directions for which I will give a brief survey.

2.1 Time and space

A rich family of timed extensions has been designed leading up to hybrid forms of process algebra. In these timed algebras some form of variable binding is used, in particular initial abstraction and integration. It has been demonstrated that large sums (alternative compositions) can be dealt with using cylindric algebras but those developments cost a price in terms of readability.

There is ample room for further development in this area. An open question is to redesign real space ACP (timed ACP in 3-dimensional space) in such a way that the equations and verifications are consistent with special relativity. After all if a

communication protocol is observed to work correctly then it should be considered correct from another inertial system just as well. Lorentz invariance should be postulated for both specifications and verifications of protocols. Having said this the next observation, however, is that ACP seems to be inconsistent with special relativity and in need for a major revision if that objective is to be met. Stated differently: ACP is about concurrency in classical mechanics only.

2.2 Mathematical results about process algebras

Non-trivial results have been obtained on the factorisation of processes in parallel components and a range of results has been obtained concerning the decidability of bisimulation equivalence and other equivalences for process notations with either recursion or with combinators that generate infinite behavior (e.g. the proper binary Kleene star, that is repetition in the way Kleene originally defined it).

2.3 Schematology and SOS

A significant development has been the schematology of congruence theorems for different SOS formats. Most simple semantic facts about ACP style process algebras can currently be derived from very general properties relating to the form of operator definitions. There is ample room for further work in this area because as it stands it is still easier to develop new operators and their axioms using explicit graph models and special purpose bisimulation definitions than via the general meta theory of SOS. However, this SOS based meta theory very much defines a stable endpoint of theory development and presentation, and for that reason it constitutes a development which will follow each design extension of the ACP family done in the old-fashioned style. Retrospective conditions with retrospective bisimulation constitute a current example of this state of affairs. Another example is found in ACP with signals.

2.4 True concurrency and data types

Non-interleaving process algebras have been designed and a significant amount of information has been developed concerning the interaction of ACP style process algebras with abstract data types. I discuss these together because it has been established that modeling true concurrency theories in ACP variants involves the introduction of the natural numbers in the form of history pointers. Other approaches to true concurrency require the introduction of localities which are also forms of data.

At face value one might expect that ACP coexists happily with abstract data types that have been specified using initial algebra semantics. But at a closer inspection that fails to be the case because ACP style axioms always need full information about equality and inequality for the class of atomic actions which serves as the most important parameter. Moreover in spite of a significant experience with combining processes and data in the specification formats PSF and μCRL, there remains to be an asymmetry: the process algebra part is quite specific about the operator sets

for processes which will be used whereas the data part is entirely liberal about the operators to be used on data. As a consequence the same facts have to be derived time and again from marginal variations of essentially the same data types and a convincing strategy for accumulating facts, definitions and operators on data types has failed to emerge. The true nature of this difficulty as well as its best solution are still hardly understood. To phrase it differently: it may be taken for granted that a family of process algebra designs leads to a reliable theory of processes, but a theory of abstract data type specification fails to deliver a theory of data. Abstract data type theory exists at the abstraction level of the schematology of SOS congruence theory mentioned above. But finding a more concrete version of abstract data type theory has proven remarkably difficult.

A deeper reason for the discrepancy between the role of processes and data in combined formalisms may be that it has become common to hide in data types complications of a kind that one would prefer not to surface in a process algebra at all. For instance finite stacks with overflow and error and even error recovery mechanisms have no counterpart in any of the ACP style process algebras. It has been said that the very concept of a stack as an abstract data type is flawed for this very reason because stacks are not the models of some well-understood equational theory.

3 How to proceed?

In all directions the development of ACP style process algebra can be pushed further to its limits if one wants to do so. Many open problems and open ends remain. The application of these techniques to protocol and system verification, either via formal equational proofs or via model checking still leaves much room for progress. That progress is likely to lead to useful applications as well.

It has become clear that the difference between ACP style process algebras and other calculi such as π-calculus and the calculus of mobile ambients is larger than one might expect. Mobile ambients seem to be so different that it cannot be understood as a feature to be designed on top of ACP or any plausible extension of it. For mobility on the other hand that is an unsolved question. Mobile features like the ones present in the π-calculus may have counterparts in ACP style, if not that is a rather interesting fact which ought to be provided with a formalization and a proof.

3.1 Program Algebra, Thread Algebra and Maurer Computers

My personal agenda is to continue with the design of algebras for programs, systems and computers very close to the lines set out for ACP but with an emphasis on different aspects. Program Algebra and its related Thread Algebra are recent outcomes of that line of work. The connection with process algebras emerges when specifications become more complex and semantic problems become harder to analyze. Indeed program algebra and thread algebra are so simple because of drastic restrictions have been made. But when composing threads, programs and machines

the resulting systems become more complicated and the simplifications of program algebra and of thread algebra sometimes become a hindrance rather than an asset, which can be removed by a transition to process algebra.

Thread algebra should find its applications in programming language semantics and moreover both in grid computing where the main form of concurrency is presented by way of multi-threading and in the theory of concurrent microprocessors where different forms of multi-threading, in particular micro-threading hold the promise of contributing to the prolonged survival of Moore's law for microprocessors.

When working on microprocessor design some theory of computers (processors, machines) is definitely needed. I have come to the conclusion that Maurer's theory of computers and computer instructions (dating from 1967 and neglected since then) provides exactly the format I need. This theory has been deliberately designed to deviate from Turing machines, and does this so drastically that an infinite parallel composition of Maurer computers, driven by multi-threading with forks is needed to simulate a single Turing machine. A detailed semantic investigation of multiple pipelined processor designs is both profitable for a systematic investigation of potential further processor speedup and for the compiler development that is needed to make use of improvements of the pipeline organization. Fortunately Maurer computers provide a perfect match with program algebra and thread algebra backed up with the useful option to cast larger system descriptions in suitable process algebras.

3.2 Grid and Web

Now we all share data via the web the next phase is to share processing via the grid. The increasing emphasis on grid computing will lead to a large number of new and complex communication and security protocols. Having made this observation it is likely that the main application area for process algebra is likely to stay within the area of protocol specification and verification where its development is supported by the most powerful tools available for proof generation, storage and validation and for model checking on increasingly large finite (and even infinite) models. The grid is so utterly incomprehensible at this stage that just this application area alone provides sufficient motivation for continued research on ACP style process algebras in my view.

4 Validity of the design rationale in 2005

I still have complete confidence in the relevance of the ingredients that together lead to ACP and similar theories. But at the same time other theories have proven very effective in quite related areas. What constitutes a breakdown of the design rationale of ACP and how would it be observed? It is the visible presence of motivating applications on the long run that decides this matter. In that regard the slow but steady build up of extensions and applications of ACP and its data type extension muCRL is reassuring and at this moment ACP style process algebra is just as convincing to me as a direction of research as it was back in 1982 when Jan Willem

Klop and I started our work in this area.